

WHAT IS CLAIMED IS:

1 1. A method for accessing data in a database table, comprising:
2 receiving a fetch request to fetch data from a base table that satisfies a query
3 predicate, wherein rows of the base table are stored in table partitions and wherein there
4 is one index partition for each determined table partition, wherein each index partition
5 includes nodes, wherein each node in each index partition includes at least one key
6 column value from a corresponding table row in the table partition associated with the
7 index partition and a location identifier identifying the corresponding table row in the
8 corresponding table partition;
9 determining a set of nodes, one from each index partition, whose key column
10 value satisfies the query predicate;
11 selecting one node from the set; and
12 returning data from the table row identified by the location identifier in the
13 selected node in response to the fetch request.

1 2. The method of claim 1, further comprising:
2 determining whether to modify a direction of the fetch request;
3 modifying the direction of the fetch request if the determination is made to
4 modify the fetch request; and
5 determining the set of nodes based on the direction of the fetch request.

1 3. The method of claim 2, wherein determining whether to modify the
2 direction of the fetch request is based on a current fetch direction and whether the current
3 fetch direction is opposite an ordering of the index partitions.

1 4. The method of claim 2, wherein modifying the direction of the fetch
2 request comprises:

1 setting the fetch direction to backward if the fetch direction is backward and the
2 fetch direction is not opposite index ordering or if the fetch direction is forward and the
3 fetch direction is opposite index ordering; and
4 setting the fetch direction to forward if the fetch direction is backward and the
5 fetch direction is opposite index ordering or if the fetch direction is forward and the fetch
6 direction is not opposite index ordering.

1 5. The method of claim 2, further comprising:
2 if the fetch request is a first fetch of the fetch request, then selecting one node
3 starting from one of: a lowest key value from each index partition if the fetch direction is
4 forward or highest key value from each index partition if the fetch direction is backward.

1 6. The method of claim 2, further comprising:
2 if the fetch request is not a first fetch of the fetch request, then determining
3 whether a previous direction of a previous fetch request is a same direction as the
4 direction of the fetch request, wherein the direction of the fetch request is capable of
5 having been modified; and
6 if the previous and current directions are different, then discarding all saved nodes
7 for the index partitions and selecting one node from a last selected node.

1 7. The method of claim 6, further comprising:
2 if the previous and current directions are the same, then scanning in the direction
3 of the fetch request from the previously saved node in each index partition.

1 8. The method of claim 1, further comprising:
2 receiving a subsequent fetch request to fetch data from the base table;
3 replacing a previously selected node selected in a previous fetch request in the set
4 with one node in the index partition including the previously selected node whose key
5 column value satisfies the query predicate to form a modified set;

1 selecting one node from the modified set; and
2 returning the table row identified by the location identifier in the node selected
3 from the modified set.

1 9. The method of claim 8, wherein the subsequent fetch request comprises a
2 fetch relative request to fetch a row that is multiple number of rows from the previously
3 selected node, further comprising:

4 performing the steps of replacing the previously selected node and selecting one
5 node multiple number of times to determine the selected node to return to the fetch
6 relative request to satisfy a fetch quantity.

1 10. The method of claim 8, wherein the subsequent fetch request comprises a
2 fetch absolute request to fetch a row that is multiple number of rows from one end of the
3 table, further comprising:

4 determining a new set of nodes, one from each index partition, by scanning from
5 one end of the index partitions for a first node whose key column value satisfies the
6 query predicate and whose key column value is greater than the previously selected node
7 if fetching forward and the key is less than the previously selected node if fetching
8 backward;

9 performing the steps of replacing the previously selected node and selecting one
10 node a number of times that is one less than the number of rows indicated in the fetch
11 absolute request to determine the selected node to return to the fetch relative request; and

12 performing the steps of replacing the previously selected node and selecting one
13 node the multiple number of times to determine the selected node to return to the fetch
14 relative request.

1 11. The method of claim 1, further comprising:

2 discarding the cached keys if the fetch request is in an opposite direction of a
3 previous fetch request;

4 determining a new set of nodes from each index partition; and
5 caching the determined new set of nodes when performing the fetch operation.

1 12. The method of claim 1, further comprising:
2 processing the fetch request to determine set of nodes in the backward direction in
3 the previous fetch request;
4 inverting the keys and sorting the inverted keys; and
5 selecting the one node containing the lowest inverted key to return.

1 13. A system for accessing data in a database table, comprising:
2 a computer readable medium;
3 a base table implemented in the computer readable medium;
4 table partitions storing rows of the base table implemented in the computer
5 readable medium;
6 index partitions, wherein there is one index partition for each determined table
7 partition, wherein each index partition includes nodes, wherein each node in each index
8 partition includes at least one key column value from a corresponding table row in the
9 table partition associated with the index partition and a location identifier identifying the
10 corresponding table row in the corresponding table partition;
11 means for receiving a fetch request to fetch data from the base table that satisfies
12 a query predicate;
13 determining a set of nodes, one from each index partition, whose key column
14 value satisfies the query predicate;
15 selecting one node from the set; and
16 returning data from the table row identified by the location identifier in the
17 selected node in response to the fetch request.

1 14. The system of claim 13, further comprising:
2 means for determining whether to modify a direction of the fetch request;

1 means for modifying the direction of the fetch request if the determination is
2 made to modify the fetch request; and
3 means for determining the set of nodes based on the direction of the fetch request.

1 15. The system of claim 14, wherein the means for modifying the direction of
2 the fetch request performs:

3 setting the fetch direction to backward if the fetch direction is backward and the
4 fetch direction is not opposite index ordering or if the fetch direction is forward and the
5 fetch direction is opposite index ordering; and

6 setting the fetch direction to forward if the fetch direction is backward and the
7 fetch direction is opposite index ordering or if the fetch direction is forward and the fetch
8 direction is not opposite index ordering.

1 16. The system of claim 13, further comprising:

2 means for receiving a subsequent fetch request to fetch data from the base table;

3 means for replacing a previously selected node selected in a previous fetch
4 request in the set with one node in the index partition including the previously selected
5 node whose key column value satisfies the query predicate to form a modified set;

6 means for selecting one node from the modified set; and

7 means for returning the table row identified by the location identifier in the node
8 selected from the modified set.

1 17. The system of claim 13, further comprising:

2 means for discarding the cached keys if the fetch request is in an opposite
3 direction of a previous fetch request;

4 means for determining a new set of nodes from each index partition; and

5 means for caching the determined new set of nodes when performing the fetch
6 operation.

1 18. The system of claim 13, further comprising:
2 means for processing the fetch request to determine set of nodes in the backward
3 direction in the previous fetch request;
4 means for inverting the keys and sorting the inverted keys; and
5 means for selecting the one node containing the lowest inverted key to return.

1 19. An article of manufacture for accessing data in a database table, wherein
2 the article of manufacture causes operations to be performed, the operations comprising:
3 receiving a fetch request to fetch data from a base table that satisfies a query
4 predicate, wherein rows of the base table are stored in table partitions and wherein there
5 is one index partition for each determined table partition, wherein each index partition
6 includes nodes, wherein each node in each index partition includes at least one key
7 column value from a corresponding table row in the table partition associated with the
8 index partition and a location identifier identifying the corresponding table row in the
9 corresponding table partition;
10 determining a set of nodes, one from each index partition, whose key column
11 value satisfies the query predicate;
12 selecting one node from the set; and
13 returning data from the table row identified by the location identifier in the
14 selected node in response to the fetch request.

1 20. The article of manufacture of claim 19, wherein the operations further
2 comprise:
3 determining whether to modify a direction of the fetch request;
4 modifying the direction of the fetch request if the determination is made to
5 modify the fetch request; and
6 determining the set of nodes based on the direction of the fetch request.

1 21. The article of manufacture of claim 20, wherein determining whether to
2 modify the direction of the fetch request is based on a current fetch direction and whether
3 the current fetch direction is opposite an ordering of the index partitions.

1 22. The article of manufacture of claim 20, wherein modifying the direction
2 of the fetch request comprises:
3 setting the fetch direction to backward if the fetch direction is backward and the
4 fetch direction is not opposite index ordering or if the fetch direction is forward and the
5 fetch direction is opposite index ordering; and
6 setting the fetch direction to forward if the fetch direction is backward and the
7 fetch direction is opposite index ordering or if the fetch direction is forward and the fetch
8 direction is not opposite index ordering.

1 23. The article of manufacture of claim 20, wherein the operations further
2 comprise:
3 if the fetch request is a first fetch of the fetch request, then selecting one node
4 starting from one of: a lowest key value from each index partition if the fetch direction is
5 forward or highest key value from each index partition if the fetch direction is backward.

1 24. The article of manufacture of claim 20, wherein the operations further
2 comprise:
3 if the fetch request is not a first fetch of the fetch request, then determining
4 whether a previous direction of a previous fetch request is a same direction as the
5 direction of the fetch request, wherein the direction of the fetch request is capable of
6 having been modified; and
7 if the previous and current directions are different, then discarding all saved nodes
8 for the index partitions and selecting one node from a last selected node.

1 25. The article of manufacture of claim 24, wherein the operations further
2 comprise:
3 if the previous and current directions are the same, then scanning in the direction
4 of the fetch request from the previously saved node in each index partition.

1 26. The article of manufacture of claim 19, wherein the operations further
2 comprise:
3 receiving a subsequent fetch request to fetch data from the base table;
4 replacing a previously selected node selected in a previous fetch request in the set
5 with one node in the index partition including the previously selected node whose key
6 column value satisfies the query predicate to form a modified set;
7 selecting one node from the modified set; and
8 returning the table row identified by the location identifier in the node selected
9 from the modified set.

1 27. The article of manufacture of claim 26, wherein the subsequent fetch
2 request comprises a fetch relative request to fetch a row that is multiple number of rows
3 from the previously selected node, wherein the operations further comprise:
4 performing the steps of replacing the previously selected node and selecting one
5 node multiple number of times to determine the selected node to return to the fetch
6 relative request to satisfy a fetch quantity.

1 28. The article of manufacture of claim 26, wherein the subsequent fetch
2 request comprises a fetch absolute request to fetch a row that is multiple number of rows
3 from one end of the table, wherein the operations further comprise:
4 determining a new set of nodes, one from each index partition, by scanning from
5 one end of the index partitions for a first node whose key column value satisfies the
6 query predicate and whose key column value is greater than the previously selected node

1 if fetching forward and the key is less than the previously selected node if fetching
2 backward;
3 performing the steps of replacing the previously selected node and selecting one
4 node a number of times that is one less than the number of rows indicated in the fetch
5 absolute request to determine the selected node to return to the fetch relative request; and
6 performing the steps of replacing the previously selected node and selecting one
7 node the multiple number of times to determine the selected node to return to the fetch
8 relative request.

1 29. The article of manufacture of claim 19, wherein the operations further
2 comprise:

3 discarding the cached keys if the fetch request is in an opposite direction of a
4 previous fetch request;
5 determining a new set of nodes from each index partition; and
6 caching the determined new set of nodes when performing the fetch operation.

1 30. The article of manufacture of claim 19, wherein the operations further
2 comprise:

3 processing the fetch request to determine set of nodes in the backward direction in
4 the previous fetch request;
5 inverting the keys and sorting the inverted keys; and
6 selecting the one node containing the lowest inverted key to return.